FINAL TECHNICAL REPORT

1 Project Identification Information

Project Title: *Mach Shared Objects*

Award Dates: from July 1, 1991 to December 31, 1994

Institution and Address: University of Utah
302 Park Building, Salt Lake City, Utah 84112

Award Number: Arpa order 8223; R&T 4331818—01
Office of Naval Research Grant N00014-91-J-4046

Program Manager: David Gunning

Program Name: Persistent Object Bases

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
SEATTLE REGIONAL OFFICE
1107 NE 45TH STREET, SUITE 350
SEATTLE WA 98105-4631

IN REPLY REFER TO:

4330
ONR 247
11 Jul 97

From: Director, Office of Naval Research, Seattle Regional Office, 1107 NE 45th St., Suite 350, Seattle, WA 98105
To: Defense Technical Center, Attn: P. Mawby, 8725 John J. Kingman Rd., Suite 0944, Ft. Belvoir, VA 22060-6218

Subj: RETURNED GRANTEE/CONTRACTOR TECHNICAL REPORTS

1. This confirms our conversations of 27 Feb 97 and 11 Jul 97. Enclosed are a number of technical reports which were returned to our agency for lack of clear distribution availability statement. This confirms that all reports are unclassified and are "APPROVED FOR PUBLIC RELEASE" with no restrictions.

2. Please contact me if you require additional information. My e-mail is *silverr@onr.navy.mil* and my phone is (206) 625-3196.

ROBERT J. SILVERMAN

2 Summary

2.1 General Overview

The commonality of Unix-based open systems has greatly advanced software portability, sharing and interoperability. The MSO project exploited that commonality in constructing two object managers (OM) layered on modern operating systems: one for C++ and one for Common Lisp programs. Both incorporate comprehensive support for persistence of all language object types, data evolution, and distributed implementation and access.

Key to the MSO approach is viewing object management as an instance of the generalized problem of accessing and combining software components. Under this view, software components as managed as pervasive system resources rather than transient language artifacts. Hence the MSO OM focuses on efficient manipulation and sharing of modules as computational resources, including system and user libraries, data schemata, class implementations, and persistent objects.

This has been achieved through OMOS (Object/Meta Object Server), which provides comprehensive module definition, combination and mapping services. These capabilities are exploitable in O-O languages through abstractions for object naming, data type management, persistence, modularity, distribution, and shared object mutation control. Compiler and platform heterogeneity are obtained through system-generated objects describing module types.

Effective utilization in Alpha_1, a large-scale computer aided geometric design (CAGD) and manufacturing (CAM) software system, constituted our primary validation criterion. This was a demanding test case, due to its code size, versatility, and existing hand-crafted methods for persistence and data evolution.

- *External publications:* [LK92]

2.2 Persistent C++

Two software prototypes were constructed providing implementations of persistent objects for C++. The run-time type information necessary for fully polymorphic manipulation of C++ objects is generated in the original prototype by a modified compiler, and in the follow-on prototype by a compiler independent preprocessor. In both systems, this run-time type information is used by polymorphic load and store functions for persistent objects.

The Alpha_1 project has replaced their Lisp-based modeling language with one based on a reuse-oriented C++ library. Central to this task is building standardized language interfaces to their geometric C++ library. Writing these interfaces is tedious and thus error prone. To avoid this problem the Alpha_1 team is experimentally incorporating goofie, the MSO dossier generator, to systematically build suitable interfaces to the geometric objects in the Alpha_1 C++ library. Since the library contains over 1,000 distinct class declarations an automated tool for interface generation is expected to dramatically reduce the effort for this project.

- *External publications:* [MCLY94]

- *PhD dissertations:* [Cla95]
- *MS thesis:* [Tha93]

2.3 Persistent Lisp

A persistent Lisp prototype (UCL+P) was developed. It fully supports Common Lisp, and provides persistence facilities for all Common Lisp data types — not just objects — via an atomic transaction facility. In addition to Lisp runtime system modifications, the UCL+P implementation relies compiler level changes to Utah Common Lisp (UCL) to efficiently and transparently manage the loading and storage of Lisp data items. UCL+P confers persistence via reachability; when a volatile entity is accessible from a persistent entity, it becomes persistent.

The changes centered around a new implementation of the Lisp reference model; instead of using direct pointers to refer to Lisp items, a double pointer scheme was used. The intermediate pointers are grouped in to a table called the Indirection Vector (IV). The IV has room for access marking bits used to track read and write access to heap resident entities. The IV also allows easy relocation of newly persistent entities into a protected region so that UCL+P may guarantee transaction semantics. Tests show that the when processing nonpersistent data, the prototype runs about 25% slower than the original Lisp system.

- *External publications:* [YSK93] [LZ93][1] [JS94] [LZ94]
- *PhD dissertations:* [Lee92] [Jac94]
- *Technical reports:* [JSK94]

2.4 Module Manipulation

A system is said to be compositionally modular if it facilitates a high degree of reuse of a simple notion of modules (namespaces) via a set of mechanisms that support effective decomposition and recomposition of programs. Etyma is an object-oriented application framework for compositionally modular systems, realized in the C++ language. The framework enables one to easily and rapidly build language processors for compositionally modular systems. An instantiation of Etyma classes essentially constitutes the internal representation of a source program in such a language. Etyma has been directly reused to build three completions: an interpreter for a module extension to the Scheme programming language, a compiler front-end for a compositional Interface Definition Language, and a compositional document processing system based on LaTeX. In addition, concepts developed in conjunction with Etyma have been reused in the specification of a layered architecture for composing object file components to construct entire applications. A description of this architecture has been awarded "Best Paper" at the 1995 International Workshop on Object-Orientation in Operating Systems.

[1] Art Lee was funded by sources other than the Mach Shared Objects project.

- *External publications:* [BL92] [BLO94] [BOL95] [BL95a]
- *PhD dissertations:* [Bra92] [Ban95]
- *Technical reports:* [BL93a] [BL93b] [BL94] [BL95b]

2.5 OMOS Server

OMOS is an active object server which manages executable programs and their components as a set of composable objects. Besides replacing the traditional services of a linker, OMOS provides dynamic object loading, shared libraries, program analysis, and custom application construction.

A shared library implementation was prototyped using the OMOS class server under the OSF/1 and HP/UX operating systems. OMOS constructs classes using descriptions that are stored in containers known as *meta-objects*. Meta-objects live within a hierarchical namespace which is exported to clients. Clients use the namespace to identify classes they wish to instantiate. The meta-object acts as a level of indirection between the client and the class implementation.

OMOS caches the implementations it generates. As a result, multiple clients instantiating the same class share the same (physical) implementation of that class. When OMOS is used as the system loader, applications that are constructed from several common pieces (libraries) automatically gain the memory sharing benefits associated with shared libraries. Since program instantiations are made through a level of indirection (provided by meta-objects), library implementations can be specialized to take advantage of the common address space layouts. Through this specialization, use of highly general and often inefficient position independent code is not required, and expensive dynamic symbol name resolution can be avoided.

The OMOS shared library implementation was ported to both microkernel and macrokernel Unix systems on the PA-RISC and Intel 386 platforms. Average efficiency improvements of 20% speedup in execution time result for short running programs invoked using OMOS.

- *External publications:* [OBLM93] [OMK93] [OM92] [BCLO93a] [OMHL94]
- *Technical reports:* [BCLO93b]

2.6 More Flexible OS Organizations

As stated above, CAGD/CAM was the driving application for the MSO approach to object management. It was soon realized, however, MSO's generalized approach applied equally well to management and manipulation of software components as objects. In particular, the OMOS technology constitutes a foundation for the development of a nanokernel-based decomposed operating system that achieves high performance while retaining inter-component protection and rich functionality. Such a system would overcome the performance/protection and performance/functionality tradeoffs that thwart traditional microkernel-based operating systems. This objective includes integration of selected research results of others, and free distribution of an unencumbered and usable version of the entire system.

This led to the formation of the the Flux project, with funding from ARPA/CSTO (contract DABT63-94-C-0058). Flux is developing an operating system that provides a much higher degree of flexibility than do traditional systems, and uses that added flexibility to circumvent the performance/functionality tradeoffs that thwart traditional highly-decomposed, microkernel-based operating systems. Foundations for this work include may MSO project themes, including (i) a comprehensive notion of modules, (ii) module manipulation cast as a system service, and (iii) a semantically enriched notion of module compatibility and adaptability. Items (i) and (ii) directly arose from the MSO project, while (iii) is a new emphasis motivated by pragmatically important inter-module concerns such as address space sharing, storage management policies, and levels of trust.

- *External publications:* [LHFL93] [FL93] [CFH+93] [FL94] [Orr95]

2.7 Web Page

www.cs.utah.edu/projects/mso

References

[Ban95] Guruduth S. Banavar. *An Application Framework For Compositional Modularity.* PhD thesis, University of Utah, August 1995.

[BCLO93a] Gilad Bracha, Charles F. Clark, Gary Lindstrom, and Douglas B. Orr. Module management as a system service. Position paper presented at OOPSLA '93 workshop "Object-Oriented Reflection and Metalevel Architectures", July 1993.

[BCLO93b] Gilad Bracha, Charles F. Clark, Gary Lindstrom, and Douglas B. Orr. Modules as values in a persistent object store. Computer Science Department Technical Report UUCS-93-005, University of Utah, January 5, 1993.

[BL92] Gilad Bracha and Gary Lindstrom. Modularity meets inheritance. In *Proc. International Conference on Computer Languages*, pages 282–290, San Francisco, CA, April 20–23, 1992. IEEE Computer Society. Also available as Technical Report UUCS-91-017.

[BL93a] Guruduth Banavar and Gary Lindstrom. A framework for module-based language processors. Computer Science Department Technical Report UUCS-93-006, University of Utah, March 5, 1993.

[BL93b] Guruduth Banavar and Gary Lindstrom. Modular language processors as framework completions. Computer Science Department Technical Report UUCS-93-026, University of Utah, October 1993.

[BL94] Guruduth Banavar and Gary Lindstrom. Etyma: A framework for modular systems. Computer Science Department Technical Report UUCS-94-035, University of Utah, October 1994. A short version was presented at the "O-O Compilation" workshop at OOPSLA '94.

[BL95a] Guruduth Banavar and Gary Lindstrom. The design of meta-architectures for object-oriented languages. In E. A. Yfantis, editor, *Intelligent Systems*, pages 363–374. Kluwer Academic Publishers, 1995. Proc. Third Golden West International Conference on Intelligent Systems, Las Vegas, NV, June 1994.

[BL95b] Guruduth Banavar and Gary Lindstrom. Object-oriented programming in scheme with first-class modules and operator-based inheritance. Computer Science Department Technical Report UUCS-95-002, University of Utah, 1995, February 1995.

[BLO94] Guruduth Banavar, Gary Lindstrom, and Douglas Orr. Type-safe composition of object modules. In *Computer Systems and Education: In honour of Prof. V. Rajaraman*, pages 188–200. Tata McGraw Hill Publishing Company, Limited, Bangalore, India, June 22-25, 1994. ISBN 0-07-462044-4. Also available as Technical Report UUCS-94-001.

THE UNIVERSITY OF UTAH

October 26, 1995

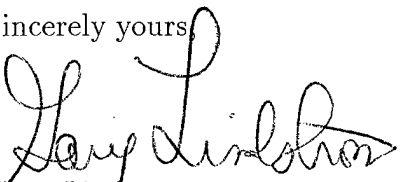
Defense Technical Information Center
Building 5, Cameron Station
Alexandria, VA 22304-6145

Dear Sir or Madam:

Enclosed please find the final technical report for grant N00014-91-J-4046 "Mach Shared Objects".

I apologize for the delay in filing this required report.

Sincerely yours,



Gary Lindstrom
Professor of Computer Science

cc: Ronald P. Moody

Phone: 801-581-5586
E-mail: Lindstrom@CS.Utah.EDU

Department of Computer Science
3190 Merrill Engineering Building
Salt Lake City, Utah 84112
(801) 581-8224
FAX (801) 581-5843

[BM93] Michael S. Blum and Robert W. Mecklenburg. Story: A hierarchical animation and storyboarding system for Alpha_1. Computer Science Department Technical Report UUCS-93-009, University of Utah, April 7, 1993.

[BOL95] Guruduth Banavar, Douglas Orr, and Gary Lindstrom. Layered, server-based support for object-oriented application development. In *Proc. International Workshop on Object-Oriented Operating Systems*, Lund, Sweden, August 1995.

[Bra92] Gilad Bracha. *The Programming Language Jigsaw: Mixins, Modularity and Multiple Inheritance.* PhD thesis, University of Utah, March 1992. Technical report UUCS-92-007; 143 pp.

[CFH+93] John B. Carter, Bryan Ford, Mike Hibler, Ravindra Kuramkote, Jeffrey Law, Jay Lepreau, Douglas B. Orr, Leigh Stoller, and Mark Swanson. FLEX: A tool for building efficient and flexible systems. In *Proc. Fourth Workshop on Workstation Operating Systems*, Napa, CA, October 1993. IEEE Computer Society.

[Cla95] Charles F. Clark. *The Dynamic Expansion of Class Hierarchy.* PhD thesis, University of Utah, August 1995.

[FL93] Bryan Ford and Jay Lepreau. Microkernels should support passive objects. In Luis-Felipe Cabrera and Norman Hutchinson, editors, *Proc. International Workshop on Object Oriented Operating Systems*, pages 226–229, Asheville, NC, December 9–10, 1993. IEEE Computer Society.

[FL94] Bryan Ford and Jay Lepreau. Evolving Mach 3.0 to use migrating threads. In *Proc. USENIX Winter Conference*, 1994. Also available as Technical Report UUCS-93-022, Computer Science Department, University of Utah.

[Jac94] James H. Jacobs. *UCL+P — A Persistent Common Lisp.* PhD thesis, University of Utah, May 1994. 117 pp.

[JS94] J. H. Jacobs and M. R. Swanson. Syntax and semantics of a persistent Common Lisp. In *Proc. of the 1994 ACM Conference on Lisp and Functional Programming*, Orlando, FL, June 1994. ACM. 10 pp.

[JSK94] J. H. Jacobs, M. R. Swanson, and R. R. Kessler. Persistence is hard, then you die! or compiler and runtime support for a persistent Common Lisp. Computer Science Department Technical Report UUCS-94-003, University of Utah, January 1994.

[Lee92] Arthur H. Lee. *The Persistent Object System MetaStore: Persistence Via Metaprogramming.* PhD thesis, University of Utah, June 1992. Technical report UUCS-92-027; 171 pp.

[LHFL93] Jay Lepreau, Mike Hibler, Bryan Ford, and Jeff Law. In-kernel servers on Mach 3.0: Implementation and performance. In *Proc. of the Third Usenix Mach Symposium*, pages 39–55, Santa Fe, NM, April 1993.

[LK92] Gary Lindstrom and Robert R. Kessler. Mach Shared Objects. In *Proceedings Software Technology Conference*, pages 279–280, Los Angeles, CA, April 1992. DARPA SISTO.

[LZ93] Arthur H. Lee and Joseph L. Zachary. Reflections on metaprogramming. Technical Report UUCS-93-004, University of Utah, May 1993.

[LZ94] Arthur H. Lee and Joseph L. Zachary. Using metaprogramming to add persistence to CLOS. In *Proc. Fifth International Conference on Computer Languages*, Toulouse, France, 1994. IEEE. Available as technical report UUCS-93-001.

[MCLY94] Robert Mecklenburg, Charles Clark, Gary Lindstrom, and Benny Yih. A dossier driven persistent objects facility. In *Proc. of the C++ Conference*, pages 265–281, Cambridge, MA, April 1994. USENIX Association. Also available as University of Utah Computer Science Department Technical Report UUCS-94-002.

[OBLM93] Douglas Orr, John Bonn, Jay Lepreau, and Robert Mecklenburg. Fast and flexible shared libraries. In *Proc. USENIX Summer Conference*, pages 237–251, Cincinnati, June 1993.

[OM92] Douglas B. Orr and Robert W. Mecklenburg. OMOS — An object server for program execution. In *Proc. International Workshop on Object Oriented Operating Systems*, pages 200–209, Paris, September 1992. IEEE Computer Society. Also available as technical report UUCS-92-033.

[OMHL94] Douglas B. Orr, Robert W. Mecklenburg, Peter J. Hoogenboom, and Jay Lepreau. Dynamic program monitoring and transformation using the OMOS object server. In *The Interaction of Compilation Technology and Computer Architecture*. Kluwer Academic Publishers, February 1994.

[OMK93] Douglas B. Orr, Robert W. Mecklenburg, and Ravindra Kuramkote. Strange bedfellows: Issues in object naming under Unix. In *Proc. International Workshop on Object Oriented Operating Systems*, pages 141–145, Asheville, NC, December 9–10, 1993. IEEE Computer Society.

[Orr95] Douglas B. Orr. Application of meta-protocols to improve OS services. In *Proc. 5th Workshop on Hot Topics in Operating Systems*, May 1995.

[Tha93] Sudheer Thakur. An empirical study of persistent object stores. Master's thesis, University of Utah Computer Science Department, Salt Lake City, Utah, December 1993. 95 pp.

[YSK93] Benny Yih, Mark R. Swanson, and Robert R. Kessler. Persistent immutable shared abstractions. In Robert H. Halstead, Jr. and Takayasu Ito, editors, *Parallel Symbolic Computing: Languages, Systems, and Applications*, number 748 in Lecture Notes in Computer Science, Berlin, November 1993. Springer-Verlag. Proceedings of the 1992 Workshop on Parallel Symbolic Computing, Cambridge, Mass.